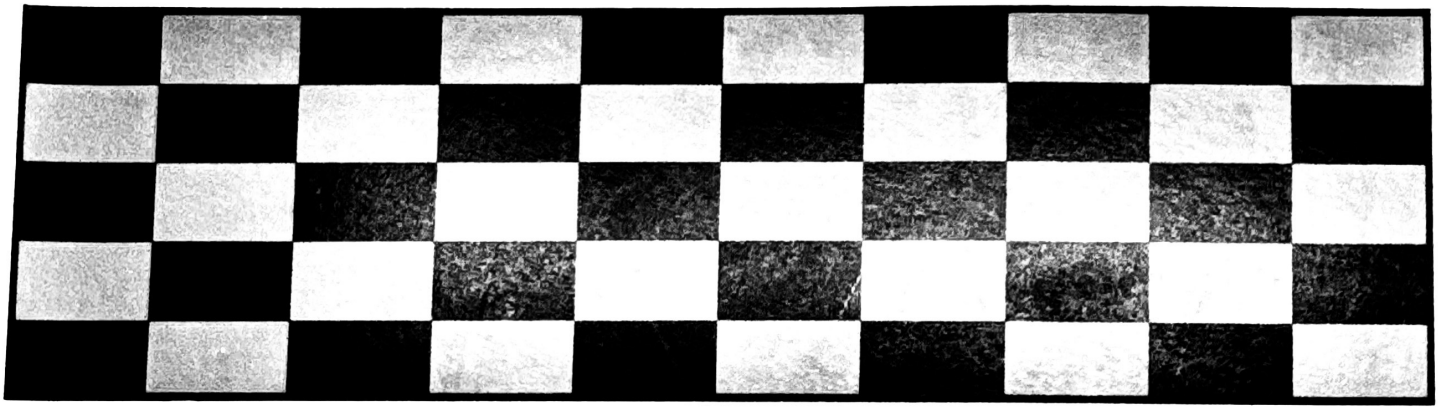
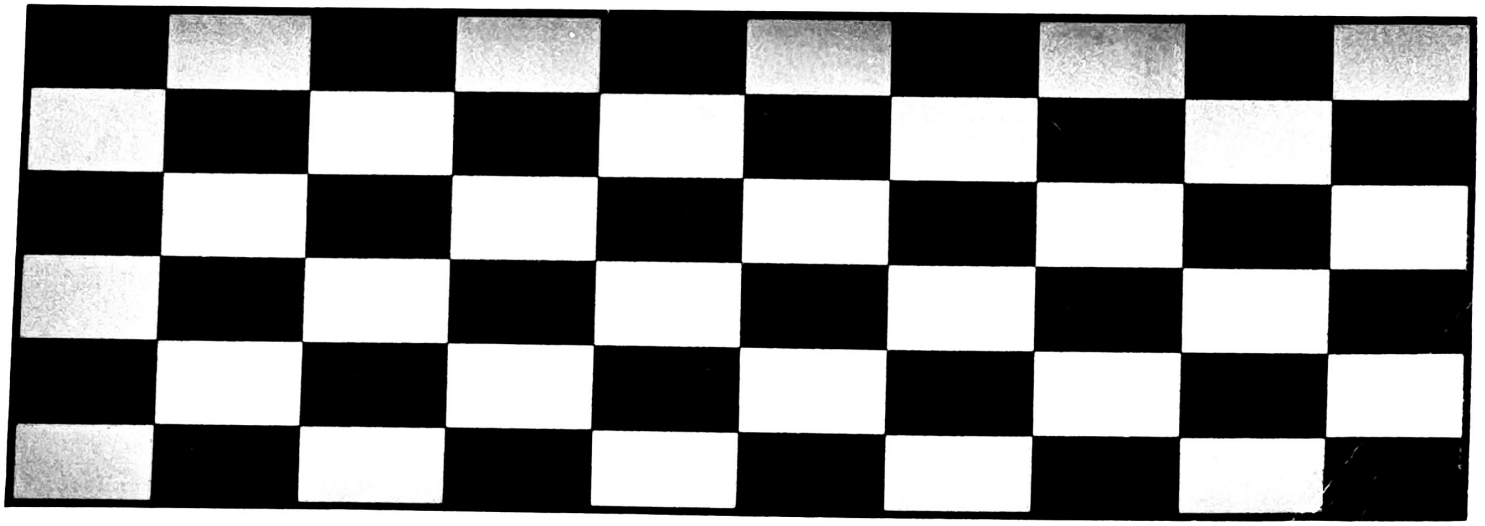
SMARTDOS

by John Chenoweth and Ron Bieber



- SMARTDOS is 100% density smart. SMARTDOS will sense the density of each disk in use and automatically reconfigure the entire system to that density.
- SMARTDOS does not require that a "system disk" has to remain in the drive, or be continually inserted and removed in order to use the DUP.SYS commands.
- With SMARTDOS you may Copy with query, (eliminates specifying each item individually).
- Counter screens – which keeps the user informed as to what the system is doing and where in the task the system is.
- Disk testing for bad or unusual sectors that may be corrected.
- RESIDUP feature allows simple yet powerful full time availability of DUP.SYS commands while leaving your program intact and ready to RUN.
- Minimum keystrokes for maximum power, e.g. a disk directory is done by pressing only one key – the drive number (great for filesearches), and "-" may be used to replace *.*.
- The ability to run from 1 to 9 autorun files sequentially.
- Built in disk drive speed check.
- SMARTDOS is only 34 single density sectors long and works with all Atari computers with a minimum of 24K RAM.



 800-228-6821
LARRY OR JIM

5230 CLARK AVENUE • SUITE 19 • LAKEWOOD, CALIFORNIA 90712 • (213) 920-8809

CHAPTER 1

INTRODUCTION TO SMARTDOS

In the years since the introduction of the ATARI computers a number of Disk Operating Systems (DOS) have appeared. Back in 1980, the only DOS available for ATARI computers was supplied by ATARI to those who bought ATARI 810 disk drives. As time passed and more people came to realize the enormous power and versatility built into ATARI computers, third-party manufacturers and software developers began to develop both new peripherals and new software for them. Today, ATARI users have a wide range of choice among disk drives and among Disk Operating Systems.

You have chosen wisely. SMARTDOS has emerged as a clearly superior Disk Operating System for ATARI computers. Not only does it permit you to get the most out of your ATARI 400, 800, or 1200XL computer, but SMARTDOS is superbly suited to the newest additions to the ATARI line, the 600XL and the 800XL, as well.

No other ATARI type DOS offers you as many powerful and easy-to-use options as those packed into SMARTDOS. And, while these features are available to those who use but a single disk drive, they become even more impressive when you add additional drives to your system. If you are one of the growing number of personal computer owners who uses his or her system for serious purposes--word processing, accounting, spreadsheet analysis, etc.--you would be well advised to consider making yours a two-drive system as a minimum. But, one drive or four, you are miles ahead with SMARTDOS.

Before delving into the many features of SMARTDOS, though, it might be wise to talk a little about what a DOS is all about and why it is needed. If you are new to computers, this discussion may help to sort out some things for you.

Computerists, apparently, just LOVE jargon. They certainly throw around enough of it. Bits, Bytes, Nybbles, RAM, ROM, CPU, 48K, 64K! The world of computers abounds in such terms. This manual will avoid computer jargon whenever possible in preference for English. When it is necessary to use a technical word or phrase, its meaning will be explained.

Just as your ATARI computer must have its own internal Operating System in order to permit its various components to work together in harmony, so, when you add a disk drive to your computer, it must bring with it another Operating System so that it can communicate with the computer (and vice-versa). While the computer's Operating System is in ROM (special memory chips inside the computer that constitute Read Only Memory), the disk drive supplies its Operating System on (what else?) a disk.

Thus, when you connect your disk drive to your computer and insert a disk containing the DOS, turning your computer on will cause the DOS to be "booted" into your computer (the term "boot" derives from bootstrap, as in pulling oneself up by one's. . .). Once the DOS has been booted, then your computer and disk drive can interact with one another freely.

Operating Systems and Disk Operating Systems consist of carefully integrated collections of machine language programs each of which is designed to perform one of the many functions or procedures built into these complex chunks of software.

GETTING STARTED

Before using your SMARTDOS disk, we strongly recommend that you read this manual in its entirety. Not only will you get a better idea as to the purpose of a Disk Operating System, but doing so may well ward off unnecessary damage to your master disk. Please include in your reading material the SOFTWARE LICENSE AGREEMENT. Not only will that inform you about the terms of your license to use SMARTDOS, but it contains vital warranty information, as well.

Caring For Your Disks

There are two methods available to you for protecting the files and programs on your disks. The easiest, and probably the safest method, is to use the write-protect tabs provided by the manufacturer of the disks. Placing one of these self-adhesive tabs over the write-protect notch (square notch) will prevent any of the files on that disk from being written to, deleted, or modified. A disk protected by a write-protect tab cannot even be formatted while the tab is in place. The other method is to protect selected files on your disks electronically. SMARTDOS permits you to PROTECT files as you choose. This method will protect files from modification or deletion, but it will NOT protect against an accidental format command.

Labeling Your Disks

To better keep track of your programs and data files, we strongly suggest that you label them carefully, using the self-adhesive labels provided with the disks. When labeling a disk, be sure to write on the label BEFORE attaching it to the disk. If you need to change a label which is already attached to one of your disks, then use only a felt tipped marking pen to make the changes. Writing on a label with a pencil or a ballpoint pen invites certain disaster for that disk.

Storing Your Disks

Your disks will, with a certain amount of care, serve you well for a considerable period of time. Please follow these suggestions for proper storage:

* Do not touch the exposed media surfaces on the disk. Handle them carefully by touching only the disk envelope.

* Do not bend or fold the disks. Remember that the disk needs to be able to spin freely within its protective envelope.

* Keep the disks in their protective sleeves when not in use.

* Store the disks vertically, as you would record albums. Do not stack them.

* Avoid storing disks near strong magnetic fields such as electric

motors. Be especially careful to keep them at least 10-12 inches away from your television set.

* Store your disks in a place that is protected from heat. Keep them away from direct sunlight.

* Liquids of any kind are not compatible with your disks, or, for that matter, with any part of your computer system. Threaten to thrash anyone who approaches within 10 feet of your system while carrying a coffee cup.

Booting SMARTDOS

For instructions on how to connect your disk drive to your computer, please consult the user's manual supplied with your drive unit. Once you have connected the drive to your computer system, then you are almost ready to boot your disk drive. At this time, Turn your drive on. Do not turn your computer on yet.

Before inserting a disk, take a minute to be sure that the "head protect" card(s) has been removed from your drive(s). These are used to protect the sensitive Read/Write/Erase head(s) of your drive(s) during shipment.

Now, insert a disk containing SMARTDOS into drive 1 (if you have but one drive, make sure it is configured to be drive 1), and turn your computer on. The red LED "busy" light on drive 1 will glow, the drive will spin, and the boot process will begin. What happens next depends on whether or not BASIC is active. If it is, then your computer will return from the boot process with a READY prompt and the cursor. If it is not active, you will be presented with the SMARTDOS menu.

A Word About ATARI Computers: In addition to the old standbys, the 400 and 800 computers, the ATARI family now includes the (discontinued) 1200XL and the brand new 600XL and 800XL computers. There may be another addition to the family in the future, the 1450XLD, but that's still up in the air. The 400, 800, and 1200XL computers use a removable BASIC cartridge, while the new 600XL and 800XL feature a built-in BASIC.

With the 400, 800, and 1200XL computers, disabling BASIC is as simple as removing the cartridge. On the newer 600XL and 800XL computers, disabling BASIC can be accomplished by pressing and holding down the OPTION key while turning on (booting) the computer. To avoid tortured descriptions in this manual, references to BASIC will be in terms of whether that language is "active" or not. Please think of those references in relationship to YOUR computer.

Now that you have booted SMARTDOS, it is time to perform an EXTREMELY IMPORTANT process: Making at least one backup copy of your SMARTDOS master disk. When you have finished the procedure outlined below, put your master SMARTDOS disk away in a safe place and do not use it for any purpose other than making new working copies. If your master SMARTDOS disk does not have a write-protect tab on it when you get it, please place one on it now.

MAKING A WORKING COPY OF SMARTDOS

The first step in making a working copy of SMARTDOS is to boot your system with the master disk. If BASIC is active, then, when you get the READY prompt, type "DOS"<RETURN> (don't include the quotes). When the DOS menu appears, remove the master disk and place a blank disk in your drive. If you have at least two drives, then leave the master disk in drive 1 and put the blank disk in drive 2. Next, type "F" (for FORMAT). The computer will respond with the prompt:

FORMAT:DISK NUMBER?

Answer by pressing the appropriate key: "1" if you have a single drive, "2" if you have two drives. Then, since FORMATting is a destructive process (it will wipe out all files on a disk), you will get another message:

PUSH "X" TO FORMAT DISK n

The "n" will be the number you typed to indicate the drive containing the disk to be formatted. If you are sure that everything is in order, then press the "X" key. The computer will proceed to FORMAT the disk. Please see the FORMAT DISK description further on in this manual for more detailed information on the formatting process.

Once your disk has been formatted, it's time to make an exact copy of your master SMARTDOS disk. To do that, press the "W" key (for WHOLE DISK COPY). The computer prompts you:

COPY WHOLE DISK
source drive # ?

If you have one drive, then answer both the source drive and destination drive questions with "1" and follow the prompts. With two drives, follow the sequence below.

Since your SMARTDOS master disk is in drive 1, answer by pressing the "1" key. Next, you will be asked for a:

DESTINATION drive # ?

Respond with a "2". The Screen will suggest that you press:

"E" TO EXIT;ANY OTHER KEY TO CONTINUE

If everything is as it should be, then press any key other than "E."

During the copy process, the screen will turn green while the DOS is reading sectors from the source disk, then red during writes to the destination disk. When finished it will let you know.

You now have a single density working copy of your SMARTDOS master disk. Please put your master disk away and use your working disk from now on. Should you wish to make a double density copy of your SMARTDOS disk, refer to p. 33 for detailed directions.

Place your new working copy of SMARTDOS into drive 1 (to avoid confusion in this manual, "drive 1" stands for the "boot" drive, whether you have one drive or several). Turn your computer off, wait a second or two, then turn it back on. This will reboot the system using the new system disk, and will serve to verify that the copying process was successful. When you have gotten into DOS and have the menu before you, press the "1" key to get a Directory of the files on the disk in drive 1. You should have the following files:

DOS.SYS
DUP.SYS
AUTORUN.SYS
RS232.ARX
DEFAULT
ARCREATE.BAS

AN OVERVIEW OF SMARTDOS PROGRAMS

THE INTERNAL FILES

DOS.SYS

This file is the core of the Disk Operating System. DOS.SYS, in addition to the routines necessary to establish and maintain communication between the disk drive and the computer, contains the File Management Subsystem (FMS), and that portion of DUP.SYS which remains in RAM memory whenever the system is operating. This RAM-resident portion of DUP.SYS is frequently referred to as Mini-DOS. Together, the FMS and Mini-DOS permit you to perform many functions from BASIC that are usually handled through the menu. These include NEW FILE NAME, PROTECT FILE, UNPROTECT FILE, DELETE FILE, and FORMAT DISK. Please read the section called XIO in Chapter 2 for details of the procedures to use (p. 18).

DOS.SYS, coupled with AUTORUN.SYS, constitutes the minimum package that must be present on a disk for it to be a SMARTDOS system disk. With these two files on it, a disk can be used to boot your system and will permit you to perform all of the normal file-handling and processing functions attendant upon programming. You will also be able to handle any of the functions mentioned in the paragraph above.

DUP.SYS

This file constitutes the Disk Utilities Package. It contains the DOS menu and routines that allow you to control those functions not supported by DOS.SYS. With DUP.SYS you may MAKE SYS FILES, COPY FILES, perform WHOLE DISK COPY, KOPY SECTORS, TEST SECTORS, RECONFIGURE/ON disk density, OBVERT RESIDUP, VERIFY/RETRY, LOAD (binary) FILES, SAVE (binary) FILES, GO TO (hexadecimal) ADDRESSes, perform SPEED CHECKs, and EXIT SMARTDOS.

Normally, you get into DUP.SYS by typing "DOS" <RETURN>. This loads DUP.SYS from disk into RAM, whereupon you are presented with the menu. When this method is used, DUP.SYS loads into the lower end of user RAM. Thus, any programs that were residing in those portions of RAM are overwritten when you load DUP.SYS. With SMARTDOS there are two ways to protect your programs. One method is to SAVE, or LIST, your program to disk before typing "DOS" <RETURN>. A faster way is for you to have OBVERTed RESIDUP. When you do that, you have

simply arranged to have DUP.SYS remain in memory. Consequently, there can be no conflict between the program you are working on and the DUP.SYS file. See the discussion on OBVERT RESIDUP in chapter 3 for further information on this powerful option.

AUTORUN.SYS

This is the file that makes SMARTDOS "SMART." When you boot your system with a SMARTDOS disk, this program transforms the Disk Operating System into the most versatile DOS available for ATARI computers.

These three programs <u>are</u> SMARTDOS. When all three are present on a disk, that disk is a full-scale SMARTDOS disk. What, then, are the remaining files for? They simply enhance an already powerful system as well as contributing further to its ease-of-use.

THE EXTERNAL FILES

These files on your system disk are referred to as "external" files because they are not an integral part of SMARTDOS. They are included in order to give you even more options than SMARTDOS gives you by itself.

RS232.ARX

This file, when run, opens a channel permitting serial devices using the RS232C protocols (modems and some printers) to be used with your ATARI computer. As it comes to you, the RS232.ARX file will not run (the "X" in .ARX stands for a numeral between 1 and 9). To implement it you must change the extender on the file to read, for example, ".AR1". If you have an ATARI 850 Interface Module and wish to use any of the serial ports in it, you would be wise to make this change. Please see the section on NEW FILE NAME in chapter 3 for instructions on how to do this.

DEFAULT

This handy program allows you to "customize" your copies of SMARTDOS to your own needs in a variety of ways. Below is a brief description of the items covered in the DEFAULT program. For a detailed description, consult the examples given in the LOAD FILEs section of chapter 3.

Disk Drive Buffers

As it comes to you, your SMARTDOS system disk assumes that you have four disk drives. Accordingly, it provides you with memory buffers (a buffer is an area of RAM memory set aside for a specific purpose, and not available to the user) to support that many drives. If you don't have that many drives you can change this section to reflect your system, thus regaining some memory.

File Input/Output (I/O) Buffers

Your system disk provides you with six of these buffers to permit normal file-handling and to support double density operation. Should you need more than six buffers you may increase the number.

Autorun File Range

SMARTDOS can, with the help of this part of the DEFAULT program coupled with the ARCREATE.BAS program described below, automatically load and run a series of BASIC programs for you. The range is from zero through nine (with zero meaning that NO autorun files will be recognized).

ARCREATE.BAS

This is the only BASIC program included on the distribution disk. To use it, type, from BASIC, RUN"D:ARCREATE.BAS". The program is self-prompting, and will permit you to assign a series of BASIC programs names having extenders ranging from .AR1 through .AR9 which will then load and run automatically when you boot with that disk.

In the chapters that follow, you will learn more about the use of the many features awaiting you in SMARTDOS. Chapter 2 describes the methods by which you may save and load files, access data, work with Input/Output Control Blocks (IOCBs), and interact with the FMS (File Management Subsystem). Chapter 3 will explore the many menu choices available to you, and will give you an idea as to how to use them.

Chapter 2

HANDLING FILES WITH SMARTDOS

A Disk Operating System allows for communication between your computer and disk drive. Without a DOS your disk drive would make a good paperweight. A good DOS must give you the ability to store and retrieve programs and files, erase old ones that are no longer needed, copy files from one disk to another, and duplicate entire disks. In addition, a DOS should permit you to save and load binary files and should handle such duties as formatting disks, producing disk directories, and facilitating your ability to rename files as you see fit.

SMARTDOS not only meets these requirements, but exceeds them. It gives you a level of control over all of the disk-related functions and procedures that is unequalled by any other DOS for ATARI computers. One unprecedented feature of SMARTDOS, for example, is that it makes your disk drives "density smart." Whenever you insert a disk into any drive in your system, the drive will automatically sense the density of that disk and adjust itself to it.

In this chapter the focus will be on those aspects of file handling that you have access to through BASIC. In Chapter 3 the DOS commands and functions available through the menu will be explored.

As you probably know, when your system is up and running and there is a "READY" prompt on the screen, that prompt tells you that your computer is ready to accept commands in BASIC. In that mode there are two ways by which you can manipulate your computer: The immediate (or direct) mode, and the deferred (or program) mode. In the former you simply type in a command without a line number. After you have done that and pressed the <RETURN> key, the computer will immediately act upon that command. In the deferred mode you precede your commands with line numbers. In this mode the computer will not act upon any of those commands until you instruct it to do so by issuing a RUN <RETURN> command.

In this discussion of the types of commands available while in BASIC, we have separated them into Program Commands and File Handling Commands. Before getting into that discussion, though, it might be wise to point out the requirements of SMARTDOS relative to the naming of programs and files.

As far as the computer (and, for that matter the disk drives) are concerned, they make no distinction between what we humans call "programs" and "files." We know that a program is a collection of sequentially numbered lines that, when run on a computer, will cause the computer to do something more or less useful. A file, on the other hand is--to us--a collection of data that we need for some reason. A mailing list is a good example of a data file.

Still, although the distinction is important to us, the requirements for naming both are identical. And, while you may think that naming files is just a convenience for us, think again. The computer needs those names as much as we do, for without them it has no way to locate them on the disk. Naming files, then, is a very important step in the process of turning your computer system into a truly useful tool.

File specifications (filespecs) consist of a file name and an optional extension, separated from one another by a period. File names (the first part of a filespec) must consist of from one to eight alphanumeric characters of which the first one must be alphabetic. That is, a file name can consist of only one character, but that character must be an upper case alphabetic character. Alternatively, a file name can consist of as many as eight characters, the last seven of which could be numeric.

ALPHANUM	WORD1
B	FILE2AB3
D032037	ZYZZYX
MAILLIST	X923A

All of these are valid file names. Notice that each of them begins with an alphabetic character. Notice, too, that there are no symbols in any of the file names. Symbols are banned from filespecs, as are spaces. If you include a space somewhere in a filespec, the computer will interpret the characters preceding the space as being the entire filespec.

Extenders added to file names can be quite handy. They permit you to further identify your files, making each as unique as you wish. Extenders may consist of from one to three alphanumeric characters. These may be all alpha, all numeric, or any combination thereof. Thus, to add some extenders to the file names used above:

ALPHANUM.AB3	WORD1.BAS
B.CDE	FILE2AB3.LST
D032037.DBD	ZYZZYX.543
MAILLIST.CLB	X923A.ASM

Once again, these are all legal filespecs. Notice the required delimiter (the period) separating the file name from the extender.

Program Commands

The commands below are used to store and retrieve programs. These commands are usually used in the immediate mode without line numbers, but they can be used within programs, too. If you do use one of these commands within a program it must be the last command. Once the computer reaches such a command it cannot return to the program.

SAVE/LOAD
LIST/ENTER
RUN

SAVE and LOAD are commands that permit you to store files to disk and to retrieve them when you need them. The SAVE command stores your files in a special way. To conserve space on your disks, the SAVE command stores your program in a "tokenized" form on the disk. A "tokenized" file is an abbreviated version of a BASIC program. Rather than saving the file as a string of ATASCII characters, one byte "tokens" are used instead to represent the BASIC commands within the program. Thus, SAVE writes a compressed version of your programs to disk.

When you wish to retrieve such a tokenized file from the disk, the command to use is LOAD. The syntax to use is similar with both SAVE and LOAD. For example:

```
SAVE"D:FILENAME.EXT"
LOAD"D:FILENAME.EXT"
```

LOAD can only be used to retrieve programs that have been SAVEd. When a program is LOADed from disk into memory, any program currently in memory will be wiped out (just as if you had typed "NEW" before issuing your LOAD command). In using any of these program commands, note that you must specify a device (D:) followed immediately with the full filespec of that file. The "D:" designation stands for Drive 1. To call up a file from any other disk requires you to specify the drive number, e.g., D2:, D3:, or D4:. The drive designation and the filespec must be enclosed within quotes.

Another way to retrieve a tokenized program file is to use the RUN command. If your intention is to LOAD a program and then RUN it, you can save a step by simply typing:

```
RUN"D2:FILENAME.EXT"
```

Your program will load and run with no further effort on your part (assuming that it was on the disk in drive 2).

LIST and ENTER are the commands to use when you wish to save and retrieve your programs in an "untokenized" form. If you have a BASIC program in memory and wish to see what that program looks like, then, in the immediate mode, type LIST. The computer will proceed to list the program for you line by line, scrolling the screen as necessary to show the entire program. This scrolling may be stopped by pressing the <CTRL> and "1" keys at the same time. <CTRL> 1 is a toggle. Pressing that combination once will stop the screen from scrolling; pressing it again will cause scrolling to resume.

With that same program in memory, if you now type:

```
LIST"D:FILENAME.EXT"
```

the computer will proceed to list the program to disk in precisely the same way it listed it to the screen; as a string of ATASCII characters. A program LISTed to disk takes up substantially more room on the disk than the same program SAVEd to the disk. Why, then, LIST a program to disk? Read on.

To retrieve a program that has been LISTed to disk requires the use of the ENTER command. This command is similar to the LOAD command except that any program currently in memory will <u>not</u> be erased to make room for the ENTERed program. Instead, the ENTERed program will merge with the program in memory. In those cases where the two programs have some program line numbers in common, the contents of the lines in the ENTERed program will overwrite the contents of the lines of the same numbers in the program in memory.

The LIST command is much more versatile than the SAVE command. With the latter you can only move a program from memory to disk. LIST, on the other hand, allows you to send a file to any peripheral device attached to your computer (except

the keyboard, of course). Should you wish a hardcopy of your program, for example, you need only type:

```
LIST"P:"
```

This will send the ATASCII version of the program directly to your printer.

Want more versatility? With LIST you can send specified lines and/or ranges of lines to the devices attached to your computer. To LIST some but not all of the lines in a program to disk, then type:

```
LIST"D:FILENAME.LST",20,80
```

This will send only the lines ranging from 20 through 80 of the named program to disk. This is a very handy way to extract subroutines from one program for use in other programs. A warning here: if you wish to LIST a segment of a program that is already recorded on the disk, then be sure to give that segment a <u>different</u> name. If you forget, and LIST that segment to disk using the original filespec, then the segment will wipe out the original file. For example, if your program is called FILENAME.LST and you wish to extract, say, lines 150, 250 to be placed on the disk, then LIST"D:SEGMENT.LST",150,250 (or whatever new filespec you choose).

LIST"P:",30,350 will list lines 30 through 350 of the program in memory to your printer. In just the same way, LIST 40,500 will list lines 40 through 500 of the program in memory to the screen. With the LIST command, the Screen Editor (device E:) is the default device.

Mainly for fun, but also to demonstrate the "device independence" of ATARI computers, LOAD or ENTER a BASIC program into memory, then type:

```
LIST"S:"
```

The designation "S:" is the device name for the Screen. The results you get may be decidedly different than those you get by LISTing to "E:". Remember that when you type LIST by itself, the computer assumes that you wish the program listed to the default device "E:".

One final note concerning the value of LIST and ENTER. If you are engaged in a programming task, then you are advised to LIST your program to disk from time to time while it is under development rather than SAVEing it, at least until the program is in its final form.

The reason for this recommendation is that when you SAVE your program in its tokenized form to disk, a "symbol table" is saved along with it. This symbol table links the variable names you have used in your program with the locations in memory containing the values for those variables. Thus, every time you LOAD that program into memory in order to do some more work on it, you load the symbol table along with it. The more you work on your program--adding new variables, getting rid of others--the longer and more congested the symbol table becomes, because, although you may have deleted some variables from your program, they won't be deleted from the symbol table. This can create quite a problem as ATARI BASIC permits you to use a maximum of 128 variables within a program. As your symbol table becomes more and more cluttered with variables, some of which you have discarded, you may well run into that limit.

The way to avoid the problem is to LIST your program to disk each time you are through working on it. The symbol table is not saved when you LIST the program to disk.

When you have finally perfected your program, that is the time to SAVE it to disk. LIST the final version of the program to disk, then type "NEW". This will clear the program (and the symbol table) from memory. Then, ENTER the program back into memory. Finally, SAVE it back to disk. Another benefit to this approach is that your SAVEd program will take up the minimum space necessary on the disk as its symbol table contains only references to variables you are actually using in that program.

There are three ways to use the RUN command. Issued by itself in the immediate mode, RUN begins execution of a BASIC program already in memory. Followed by a drive number and a filespec, as, for example:

```
RUN"D:FILENAME.EXT"
```

and you tell the computer to LOAD a file, and when LOADed, proceed to RUN it. Thirdly, you may use the RUN command within a program. If you have a program made up of several segments which need to be run in sequence, then you can make the last line of each segment read:

```
500 RUN"D:SEGMENT.002"
```

In this example, of course, the "500" stands for the program line. Please note that when you use this approach, each of the segments "chained" together must be able to stand and run on its own. Programs cannot pass variables to one another.

File Commands

```
OPEN/CLOSE
GET/PUT
INPUT/PRINT
NOTE/POINT
XIO
```

These commands are generally used in the deferred (program) mode. To emphasize the point, the examples below will be preceded by line numbers.

In order to understand the business of file handling, it is necessary to begin with a discussion of Input/Output Control Blocks. An Input/Output Control Block (IOCB) is, in effect, a channel that connects the computer to any one of several possible devices. Your computer establishes eight of these "channels" or "device links" every time you boot it. Three of them are automatically OPENed by and reserved for the computer's use. IOCB #0 connects the keyboard to the editor. The keyboard (device K:) is, after all, just another peripheral to the computer, and the editor (device E:) is another. Connecting these two devices together through IOCB #0 is what permits you to see on the screen (device S:), what you have typed on the keyboard and sent to the editor.

IOCB #6 connects the computer to the screen (S:) so that data can be displayed appropriately in the graphics modes. Perhaps a short program will illustrate the principle:

```
10 GRAPHICS 2:POKE 752,1
20 PRINT:PRINT"And This through E: (IOCB #0)";
30 PRINT:PRINT #6;"this went through S:"
40 PRINT:PRINT #6;"   (IOCB #6)"
50 GOTO 50
```

When you RUN this little program, most of the screen turns black, which is the default color for GRAPHICS 2. But down at the bottom there is a four line band of blue. That is the "text window." It is still in Graphics 0 (the text mode). Thus, if you want to display messages in both, you must know where to send the messages. A normal PRINT statement, followed by a message in quotes, will be routed by the computer through IOCB #0 to the editor and thence to the "normal" part of the screen (the text window).

To send a message to the GRAPHICS 2 screen requires you to specify where it is to go. Thus, the "PRINT #6;" statement: "#6" refers to IOCB #6.

Two more brief comments on the program. the POKE 752,1 command simply turns off the cursor (in the text window), and is thus a cosmetic touch. Line 50 reads: "50 GOTO 50." All that does is to put the computer into an endless loop to keep the program from ending. Were it to end normally, the message in the text window would be scrolled up out of the text window in order to display the READY prompt. Remove the line to see what happens.

IOCB #7 is reserved by the computer for commands such as SAVE, CLOAD, and LPRINT.

What all of this leaves the programmer with is five IOCBs which are unequivocally available for use (IOCB #1 through IOCB #5). If you are careful, there are occasions when you can also use IOCB #6 and IOCB #7, too.

OPEN/CLOSE

If you wish to use an IOCB, then you must OPEN one (with the exception of the three mentioned above). The format of an OPEN statement can be illustrated by an example:

```
100 OPEN #1,8,0,"D:FILENAME.EXT"
```

This statement OPENs IOCB #1 and prepares the channel to WRITE information to a file named "FILENAME.EXT" on drive 1. In the example above, the "#" symbol is mandatory, the "8" dictates a specific function, the "0" is a required byte (the circumstances when it would be other than "0" are not worth mentioning), and the last paramenter is a device specification, such as "S:", "E:", "P:", "C:", "K:", "R:", or a filespec complete with a drive designation.

The values available for use in the second parameter (the "8" in the example above) determine what the OPENed channel will do:

4 - READ: will read from the beginning of the specified file

6 - READ the disk directory as indicated by the filespec

8 - WRITE: Writes will start at the beginning of the specified file, and will overwrite any data already in the file. This option creates a file on the disk bearing the filespec you specify, or, sends data to another output device, such as a printer.

9 - APPEND: Will add data beginning at the end of the specified file

12 - READ and WRITE relative to specified file. Begins at start of file

In the example above, IOCB #1 was linked to the disk drive. It could have been linked to other devices, as well. Some examples:

```
100 OPEN #1,8,0,"P:"
100 OPEN #1,8,0,"C:"
```

By now you probably know that "P:" is the printer, and "C:" is the Program Recorder (cassette). Note that you can write to the printer, but you can't read from it! In other words, don't try:

```
100 OPEN #1,4,0,"P:"
```

unless you like to be frustrated. When you work with IOCBs, just remember that what you are trying to do must make sense. Just as you wouldn't try to read from the printer, so you shouldn't try to write anything to the keyboard. Please see p.15 for a program that will demonstrate these principles.

Once opened, a channel remains open until a program ENDs. An END statement closes all channels. However if you press the <BREAK> key, or if there is a STOP statement in the program, or an error is encountered, all open channels will remain open. To be on the safe side, therefore, you should always explicitly CLOSE channels when you are through with them. The format is:

```
100 CLOSE #X
```

Each open channel must be closed by a CLOSE #X command, where "X" stands for the number of the channel to be closed. The only general CLOSE statement is END.

GET/PUT

Once you have OPENed a channel, what can you do with it? For one thing, you can GET individual bytes of data from a device, or PUT individual bytes to a device. In most cases bytes will be PUT to disk files or retrieved from disk files (GET). However, don't forget the device independence ATARI computers are famous for. Try this little program as an illustration of the point:

```
10 GRAPHICS 2:POKE 752,1
20 POSITION 6,3
30 FOR LOOP=0 TO 7
40 READ BYTE
50 PUT #6, BYTE
60 FOR WAIT=0 TO 200:NEXT WAIT
```

```
70 NEXT LOOP
80 PRINT:PRINT "    The SECRET message revealed!!"
90 PRINT:PRINT "    PRESS <SYSTEM RESET> TO QUIT!"
100 GOTO 100
110 DATA 83,109,225,210,84,100,239,211
```

PUT and GET will deal with byte-sized chunks of data, one at a time. Since the largest number that a byte can represent is 255, the range of values that can be used by the GET/PUT commands is 0-255, or 256 different values. Data PUT to disk can only be retrieved by GETting them.

One interesting use of the GET function is to have it give you a "live" keyboard. If, somewhere in a program, you need to ask the user for a single key response, you can use the GET function to relieve the user of the necessity to press the <RETURN> key. Try this:

```
10 OPEN #1,4,0,"K:"
20 OPEN #2,8,0,"E:"
30 POKE 752,1
40 POSITION 4,7
50 PRINT "PRESS ANY KEY (JUST ONE, PLEASE)"
60 GET #1,Z
70 POSITION 5,10
80 PRINT "YOU HAVE PRESSED THE '";:PUT #2,Z
90 PRINT "' KEY."
100 CLOSE #1:CLOSE #2
110 OPEN #1,4,0,"K:"
120 PRINT:PRINT "       WANT TO TRY AGAIN (Y/N)"
130 GET #1,Y:CLOSE #1
140 IF Y=ASC("Y") THEN GOTO 10
150 IF Y<>ASC("N") THEN GOTO 110
160 PRINT CHR$(125):POKE 752,0
```

Notice that when you run the program, you never have to press the <RETURN> key. That is what is meant by a "live" keyboard. While the program above is not particularly elegant, studying it (and the other sample programs in this manual) will reveal a wealth of information about programming techniques. For one thing, it is pretty well error trapped: That is, there is very little the user can do to make the program "crash," except by pressing the <BREAK> or <SYSTEM RESET> keys.

INPUT/PRINT

As with most of the commands in ATARI BASIC, these two have several purposes. Used by themselves in a program, they perform one set of functions. Used in conjunction with the OPEN/CLOSE commands, and they perform quite differently.

When you encounter a line in a program such as:

```
50 PRINT "THIS IS A MESSAGE"
```

then, when the program is RUN and that line is processed, the computer will PRINT the string of characters between the quotation marks through IOCB #0 to the screen of your monitor. On the other hand, when these lines are present in a program:

```
100 OPEN #1,4,0,"D:FILENAME.EXT"
110 OPEN #2,8,0,"E:"
120 INPUT #1,A$
130 PRINT #2;A$
140 CLOSE #1:CLOSE #2
```

the INPUT command will proceed to get a block of data (up to 110 characters) from the disk file named FILENAME.EXT, and the PRINT statement will print that string (through IOCB #2) to the screen.

As you can see, INPUT and PRINT deal with blocks of data--as opposed to the byte-by-byte limitation of the GET/PUT functions. With INPUT from a disk file, the string obtained by the function will include all characters up to an End of Line (EOL) character. In an analogous manner, an INPUT statement in a program will accept any characters you type in response to the request, up to the point where you press the <RETURN> key. Suppose you are running a program which asks you to type in your name. The relevant program lines would be:

```
10 DIM NAME$(30)
--
--
--
110 PRINT "PLEASE ENTER YOUR NAME (FIRST, INIT., LAST)";
120 INPUT NAME$
```

When you respond to this request you would type:

Garfine Q. Pendragon, Esq. <RETURN>

NAME$ would then consist of that entire string, including the periods, spaces, and the comma. The end of the string is determined by the EOL character (the <RETURN> key).

There is a mild warning due with respect to the PRINT statement. You must watch your punctuation. If you type the following in a program:

```
500 OPEN #2,8,0,"E:"
--
--
--
650 PRINT #2,NAME$
```

en you can expect the string printed on the screen to start printing at the first bed position. Confused? Just remember the functions of commas and semicolons en they are used in PRINT statements--commas cause tabbing; semicolons catenate. To be on the safe side, therefore, you are urged to use semicolons in NT statements, expecially those used in file handling. The format is:

650 PRINT #2;NAME$

rove this to yourself, type in the program listed in this manual under the ng of WILDCARDS in Chapter 3. Line 80 of that program reads: 80 PRINT . RUN the program in its original form first, and note where the listing of

filespecs appears on the screen. Then, change line 80 to read: 80 PRINT #2,A$. RUN it again and see the difference.

When you use commas in PRINT statements to a disk file, the records PRINTed will not start at the beginning of the segment assigned for that record, but will begin eleven bytes into the segment. If you later INPUT that record and proceed to PRINT it to the screen with a PRINT statement containing a comma, it will print at the twenty-first position on the screen (two tabs in from the left). In other words, you would be compounding the problem.

NOTE/POINT

Normally, disk records are stored in a sequential fashion within a file. While that is a simple and quick way to store the records, it can cause problems in accessing them later. If you had to access those records in a sequential fashion the search time would probably discourage you from the task. If you wanted to examine the last record in the file, the computer would have to work its way through all the intervening records until it reaches the one you want. How much more efficient it would be if you could access your records in a random fashion, thus allowing direct access to any record in the file, regardless of its position.

NOTE and POINT permit such direct, random access. NOTE makes a note of the sector and byte that mark the beginning of a record. POINT moves the read/write/erase head of your disk drive to that sector, and instructs it as to which byte to read, or write, in that sector.

There are many different search methods available; some more efficient than others. The sample program below is a very simple example of an "indexed" search routine. The program opens two files. One file is for the data you intend to store, and the other is used to provide an "index" which tells the disk drive where to begin to look for the records in the data file.

```
100 DIM SECTOR(10),BYTE(10),NAME$(30),
ADD$(30),CSZ$(30),PHONE$(15)
110 OPEN #1,8,0,"D:ADDFILE":OPEN #2,8,
0,"D:NPUPDATE"
120 PRINT CHR$(125):POSITION 2,4:PRINT
"ENTER NAME:";:INPUT NAME$
130 IF NAME$="" THEN 500
140 POSITION 2,6:PRINT "ENTER ADDRESS:
";:INPUT ADD$
150 POSITION 2,8:PRINT "ENTER CITY,ST,
ZIP:";:INPUT CSZ$
160 POSITION 2,10:PRINT "ENTER PHONE
NUMBER:";:INPUT PHONE$
170 NOTE #1,SEC,BYT
180 PRINT #1;NAME$
190 PRINT #1;ADD$
200 PRINT #1;CSZ$
210 PRINT #1;PHONE$
220 PRINT #2;SEC
230 PRINT #2;BYT
240 IF REC>10 THEN 500
250 REC=REC+1
```

```
260 GOTO 120
500 CLOSE #1:CLOSE #2
520 OPEN #1,4,0,"D:ADDFILE":OPEN #2,
4,0,"D:NPUPDATE"
530 FOR I=1 TO REC:INPUT #2,SEC:
SECTOR(I)=SEC
540 INPUT #2,BYT:BYTE(I)=BYT:NEXT I
550 PRINT CHR$(125):PRINT "PRINT
RECORDS":POSITION 2,6:PRINT "ENTER
RECORD NUMBER ";:INPUT REC
560 SEC=SECTOR(REC):BYT=BYTE(REC)
570 POINT #1,SEC,BYT
580 INPUT #1,NAME$
590 INPUT #1,ADD$
600 INPUT #1,CSZ$
610 INPUT #1,PHONE$
620 PRINT NAME$:PRINT ADD$:PRINT CSZ$:
PRINT PHONE$
630 FOR W=1 TO 600:NEXT W:GOTO 550
```

While this program will permit you to create a very short mailing list (10 records), it is NOT very practical as it stands. Since the section of the program to create the records is connected to the section on reading them, every time you **RUN** the program it will wipe out all of your previous records!! If you have created a set of records by running the program once, and later want to review those names and addresses again, then LOAD the program. When it is loaded, do not **RUN** it. Instead, type "GOTO 500". That way, the program will begin execution at the section designed to read the records.

XIO

The XIO command is a general purpose Input/Output command with many uses. Because of space limitations, this manual will only discuss the functions of XIO that permit you to do things from BASIC normally done from the DOS menu. For further information on this powerful BASIC command, please consult the appropriate books listed in the Bibliography.

As was noted in Chapter 1, the XIO command permits you to use the menu functions **NEW FILE NAME**, **PROTECT FILE**, **UNPROTECT FILE**, **DELETE FILE**, and **FORMAT DISK**. The procedures to use are:

```
NEW FILE -- XIO 32,#1,0,0,"D:OLDFILE,NEWFILE"
PROTECT  -- XIO 35,#1,0,0,"D:FILENAME.EXT"
UNPROTECT-- XIO 36,#1,0,0,"D:FILENAME.EXT"
DELETE   -- XIO 33,#1,0,0,"D:FILENAME.EXT"
FORMAT   -- XIO 254,#1,0,0,"D1:"
```

These procedures may, and in fact should, be used in the immediate mode, rather than from within programs.

CHAPTER 3

MENU CHOICES

While the menu of SMARTDOS may appear confusing at first, with its welter of choices, you will soon discover that SMARTDOS is one of the easiest to use disk operating systems available for ATARI computers. In fact, you may wonder why it took so long for such a powerful and versatile system to appear.

In order to make this manual as easy to use as possible, the menu choices in SMARTDOS will be dealt with in the order in which they appear on the menu. Wherever a feature of SMARTDOS can be used with several menu items, as with "wildcards," that feature will be described in detail first. Thereupon the application and use of that feature will be described in conjunction with each of the menu items to which it applies.

When you boot SMARTDOS and are presented with the menu (if BASIC is active you will need to type DOS <RETURN> to see the menu), the box near the top of the screen provides you with much information about your system. On the top line within the box the density status (and availability) of each of your disk drives is displayed. The next line informs you of the present status of RESIDUP (see pp. 34-35 for an explanation), the amount of RAM available to you (called BUFFER), and the current status of VERIFY (see p. 35).

Beneath these status lines are displayed the array of choices available in SMARTDOS.

DRIVE #=FILE LIST <+OPTION=PRINT>

To obtain a directory listing of the disk in any drive, simply press the number corresponding to that drive. There is no need to specify a device, just the number. If, in addition, you wish a hardcopy listing of that disk's directory, then press and hold the OPTION key and press the number. Be sure your printer is on-line.

When you press the number of the drive, you will be presented with a substantial amount of information. The drive number and its density status (SNG or DBL) is reported along with the filespecs, including extensions where appropriate, of all the files on that disk, and the number of sectors each takes up on the disk. Finally, you are told the number of free sectors remaining on the disk.

MAKE SYS FILES

After you have formatted a disk, the next step is to write DOS.SYS and DUP.SYS to that disk. As the amount of space taken up by these two files is moderate (especially in double density), we recommend that you make it a habit to include these SYStem files on most of your disks.

If your system has been booted under SMARTDOS and you wish to write the SYStem files to a disk, then press "M". The screen will display the message:

MAKE SYS FILES:DISK#?(OPTION=NO DUP)

MAKE SYS FILES WITH DOS.SYS AND DUP.SYS

Insert a formatted disk into your drive and press "1" (place the disk in drive 2 if you have more than one drive and press "2"). The screen will prompt you with:

PUSH "Y" TO WRITE DOS & DUP:DISK n

Press "Y" to begin the process. The screen prompt will inform you that it is:

WRITING NEW DOS & DUP:DISK n

Upon completion, the prompt will read:

PRESS CHOICE OR <RETURN> FOR MENU

MAKE SYS FILES--DOS.SYS ONLY

There may be times when you will want to have disks that will boot, but which will leave as much space as possible for your programs and files. One way to do this is to leave DUP.SYS off of those disks. DUP (for Disk Utilities Package) is necessary only if you want to perform any of the functions supported by that set of programs. To write DOS.SYS to your formatted disk without including DUP.SYS, hold down the OPTION key and press "1" to write DOS to the disk in drive 1 (or "2" if you have more than one drive). The screen will remind you that it is:

WRITING NEW DOS ONLY :DISK n

When finished, the screen will display the message:

PRESS CHOICE OR <RETURN> FOR MENU

A word of CAUTION is in order here. Before you write SMARTDOS to any disks which already contain programs and/or files, be sure they are compatible with SMARTDOS. One safe way to accomplish this is to boot your system with SMARTDOS, then load each of the files from the program/file disk to assure yourself of compatibility. Files and programs saved under DOS 1, for example, may cause some problems. Once you are satisfied, proceed to write DOS & DUP (or DOS alone, if you prefer) to the disk.

WILDCARDS

Just as many card games feature particular cards which can substitute for the values of other cards, so SMARTDOS contains wildcards to enhance both its power and its ease of use. As with any powerful tool, wildcards need to be handled carefully lest they get out of control and do things you didn't intend.

SMARTDOS provides three wildcards for your use:

? * =

The question mark (?) replaces any single character in a filespec, either in the name or the extension, or both. Perhaps you have the following files on a disk:

ANSWER.BA0

TEST.BA1
EXAM.BA2
QUIZ.BA3
FLUNK.BA4

In this example, your computer is in BASIC mode, and you don't want to go to the menu to check the directory because you know that you want the FLUNK file, but you can't remember what BA number you assigned to it. All you need do is to specify, for example, LOAD"D:FLUNK.BA?" and the wildcard will find the right file for you.

Or, consider the following files:

CLANK.BAS
BLANK.LST
FLANK.SKN
SLANK.ASM

You want the file with the SKN extension, but are fuzzy about the first letter of the file name. Try (from BASIC) RUN"D:?LANK.SKN" and your file will load and run.

The asterisk (*) stands for any group of letters and/or numbers in a filespec up to and including the entire file name or the whole extension. A file named "ASTERISK.BAS" could be successfully called up by "D:*ISK.BAS" or "D:*ISK.*" or even "D:ASTE*.*". There are other possibilities, too.

The equals sign (=) represents entire filespecs--names and extensions. This facility can be particularly useful in conjunction with the COPY FILE, PROTECT FILE, UNPROTECT FILE, and DELETE FILE menu choices (see below).

WILDCARDS may be used with any of the following menu choices:

COPY FILE
NEW FILE NAME
PROTECT FILE
UNPROTECT FILE
DELETE FILE
LOAD FILE

Before leaving the subject of wildcards, you should know that they have a powerful use outside of their menu applications. Wildcards may be used from BASIC, as well. Some of the examples above demonstrated that. Although it is beyond the scope of this manual to describe all of the various ways in which wildcards may be used in programming and file handling, bear in mind that wildcards can be worth their weight in gold in many programming situations. To remind you of that fact, type the following program into your computer EXACTLY AS SHOWN. When you have finished typing it, don't RUN it yet. Instead, SAVE it to disk first. This is always good practice in case it turns out you have made a mistake and the computer "locks up" when you run it.

```
10 REM ** D:BASICDIR.BAS
20 DIM A$(790),DIRNO$(7)
30 GOSUB 200
40 OPEN #1,6,0,DIRNO$
50 OPEN #2,8,0,"E:"
```

```
60 TRAP 100:FOR FILESPEC=1 TO 64
70 INPUT #1,A$
80 PRINT #2;A$
90 NEXT FILESPEC
100 TRAP 40000
110 POKE 752,0:END:REM "END" CLOSES FILES
200 POKE 752,1
210 PRINT CHR$(125):REM CLEAR THE SCREEN
220 OPEN #3,4,0,"K:"
230 POSITION 3,5:PRINT "EXAMINE WHICH DRIVE? "
240 GET #3,Z:Z=Z-48:PRINT Z
250 CLOSE #3
260 IF Z<1 OR Z>2 THEN GOTO 210
270 IF Z=1 THEN DIRNO$="D1:*.*":RETURN
280 IF Z=2 THEN DIRNO$="D2:*.*":RETURN
```

This simple BASIC program will permit you to examine the directory of any disk without leaving BASIC! Notice that DIRNO$ is defined using the asterisk wildcard so that all the files on the disk will be displayed on the screen.

COPY FILE

The COPY FILE function of SMARTDOS will allow you to copy files in a variety of ways. You may copy a file from one drive to another, or copy a file UNDER ANOTHER NAME to the same disk, or copy a file from one disk to another using the same drive.

In its simplest use, COPY FILE will permit you to make backup copies of important files on separate disks. One of the best ways to reduce frustration and avert disaster is to get into the habit of regularly backing up your important disks and files. In this light, COPY FILE may well become a familiar friend among the menu choices in SMARTDOS.

To copy a file whose name you know from one disk to another, type DOS <RETURN> to get back to the menu. When the menu appears, press "C" and the message will read:

COPY:SOURCE,DEST

Suppose that your file name is "STARS3D.ANA" and it is on the disk in drive 1. To copy that file from drive 1 to drive 2, just type "C", then:

STARS3D.ANA,2:STARS3D.ANA <RETURN>

Notice that it was unnecessary to specify a drive number in front of the first filespec, as the file was on the disk in drive 1, the default drive. Next, notice that it was only necessary to specify a number and a colon for drive 2, along with the filespec. There is a shortcut method for doing what we just did, too. First, type "C", followed by:

STARS3D.*,2: <RETURN>

The secret here is to use a wildcard somewhere in the first use of the filespec.

When you do that, you enable SMARTDOS to assume that you want the same filespec attached to the copy written to drive 2. You could just as easily have typed "*.ANA,2:", and accomplished the same thing.

In these examples, the file was copied to the disk in drive 2 under the same name as it had on drive 1. Note that you may save it under an entirely different name. For example:

STARS3D.ANA,2:STARS3D.BAK <RETURN>

would have worked just as well. The BAK extension is often used to indicate BAcKup file copies.

Should you wish to make a copy of a file on the same disk but under a different name, then, assuming that you have a file named "GOLD", which is on drive 1, type "C", followed by:

GOLD,BRASS <RETURN>

Again, notice the absence of any drive numbers. In the example, the file was on a disk in drive 1 and you wanted to copy it to the same disk under a different name.

CAUTION: Please use great care when exercising this option. The DOS will let you attach the same name to any number of files on the same disk, but if you do you won't be able to access any of them but the file of that name which appears first in the directory.

Should you desire to copy a file from one disk to another, but using only one drive, then again assume drive 1, and a file named "GRAPHICS.DEM". To copy that file using a single disk drive, then type "C", followed by:

GRAPHICS.DEM <RETURN>

By invoking the name just once you signal SMARTDOS that this is to be a single drive copy and you will be prompted as to when to insert your Source disk and your Destination disk. Wise practice here is to place a write-protect tab on your Source disk in order to stave off disaster should you get confused during the process. Long files sometimes require several disk swaps, and it is much too easy to lose track of where you are. If you insert your Source disk by accident when you are asked to insert your Destination disk, you will get an ERROR 144 (device done error) and the operation will be aborted, but at least your source disk will still be intact.

USING WILDCARDS WITH COPY FILE

Wildcards and COPY FILE were meant for each other. The judicious use of wildcards adds tremendous power and flexibility to many of the menu choices, but perhaps none more clearly than COPY FILE.

Consider the following files:

FARKLE.BAS
GLSMNG.BAK
T14287.NUM

DONEIT.BAS
OVERALL.ASM
TOOMUCH.BAS

To copy all of the files bearing the extension BAS from a disk in drive 1 to a disk in drive 2, type "C", and:

*.BAS,2: <RETURN>

All of the BAS files will be duly copied under their full original names onto the disk in drive 2, while none of the other files will be affected (or copied). A reminder: Whenever you have used a wildcard in the Source drive specification, you need specify only the destination drive number and the colon--SMARTDOS will figure the rest out for you.

Another way by which wildcards can enhance the copy function is to use them to copy all of the files on one disk to another disk. The only file COPY FILE will not copy to disk is DOS.SYS. To write DOS.SYS you must use the MAKE SYS FILES ("M") option, or the WHOLE DISK COPY ("W") option.

Wildcards will not copy any files bearing the extension SYS. Any other file, regardless of extension designation, will be copied. A procedure to use to copy all non-SYS files from one disk to another (in different drives) is to type "C", then type:

.,2: <RETURN>

Notice here that, once again, the Source drive was drive 1 and didn't need to be specified.

An even easier way to copy all non-SYS files from one drive to another is to type "C" then type:

=,2: <RETURN>

If you will recall from the earlier explanation of wildcards (beginning on p. 20), the asterisk may stand for groups of characters, either in the filename or the extension, or both. The equals sign, on the other hand, stands for the entire filespec--both the name AND the extension. Thus, in the example above, to copy all non-SYS files from drive 1 to drive 2, you need only type the four characters shown.

Still another way by which you can copy selected files from one disk to another is to use the "Query" (/Q) function. If you are not sure which of the files on a disk you wish to copy to another disk, then type "C", and:

=/Q,2: <RETURN>

This sequence tells the computer that you want to examine all the files on the disk in drive 1, and to have it query you before copying each to a disk in drive 2. As the computer gets to each file, it displays the name and asks whether you want it copied. A "Y" response will cause the file to be copied, and an "N" will cause the computer to skip to the next file.

The default (normal) mode of the Query function is "off." To turn it "on" ires you to append the "/Q" key sequence to the end of the instructions for the :e drive. Look again at the example above. The "=" stands for "examine and ay all files, with all extensions, that are to be found on drive 1." When you o that "=" sign the "/Q" sequence; the message is added, "and ask the user if she really wants them copied."

One final note about the COPY FILE function. When you are copying all files ose selected by the Query option), the Destination drive number and the colon l that are required. Thus, "=/Q,2:" is understood by SMARTDOS to mean "copy lected files under their full, original names to drive 2."

FILE NAME

This menu option permits you to rename files on any of your disks. It may be with or without wildcards. If you choose to use wildcards with the NEW FILE menu function, then please use great care to keep the WILDcards properly

To rename a single file, type "N" and:

RAINBOW.TMP,RAINBOW.BAS <RETURN>

uld you want a more drastic change:

RAINBOW.TMP,POTOGOLD.CMD <RETURN>

The procedures for renaming a group of files are similar to those for copying of files. Suppose that you have a number of files on a disk which share a file extension, such as TMP (which usually stands for TeMPorary). Your s to change the extension on all those files to something else, perhaps BAS. , type "N", and:

.TMP,.BAS <RETURN>

ll change the extension on all files which had the TMP extension to BAS disturbing the filespecs of any files on the disk that did not contain the ension.

nother possibility would be to eliminate file extensions. If you use a group frequently you may tire of typing the entire filespec, extension and all. To f the extensions, type "N", and:

.BAS, <RETURN>

names will be preserved, but the extensions, BAS in this example, will be In this instance, the asterisk stands for the file name(s), but not the

ns.

imple. To add a common

The default (normal) mode of the Query function is "off." To turn it "on" requires you to append the "/Q" key sequence to the end of the instructions for the Source drive. Look again at the example above. The "=" stands for "examine and display all files, with all extensions, that are to be found on drive 1." When you add to that "=" sign the "/Q" sequence; the message is added, "and ask the user if he or she really wants them copied."

One final note about the COPY FILE function. When you are copying all files (or those selected by the Query option), the Destination drive number and the colon are all that are required. Thus, "=/Q,2:" is understood by SMARTDOS to mean "copy all selected files under their full, original names to drive 2."

NEW FILE NAME

This menu option permits you to rename files on any of your disks. It may be used with or without wildcards. If you choose to use wildcards with the NEW FILE NAME menu function, then please use great care to keep the WILDcards properly tamed.

To rename a single file, type "N" and:

RAINBOW.TMP,RAINBOW.BAS <RETURN>

Or, should you want a more drastic change:

RAINBOW.TMP,POTOGOLD.CMD <RETURN>

The procedures for renaming a group of files are similar to those for copying groups of files. Suppose that you have a number of files on a disk which share a common file extension, such as TMP (which usually stands for TeMPorary). Your intent is to change the extension on all those files to something else, perhaps BAS. To do so, type "N", and:

.TMP,.BAS <RETURN>

This will change the extension on all files which had the TMP extension to BAS without disturbing the filespecs of any files on the disk that did not contain the TMP extension.

Another possibility would be to eliminate file extensions. If you use a group of files frequently you may tire of typing the entire filespec, extension and all. To get rid of the extensions, type "N", and:

.BAS, <RETURN>

The file names will be preserved, but the extensions, BAS in this example, will be purged. In this instance, the asterisk stands for the file name(s), but not the extensions.

Adding extensions to file names is just as simple. To add a common extension, such as CMD to a group of files, type "N", and:

,.CMD <RETURN>

This will add the CMD extension to ALL files on the disk which do not currently have extensions. Be careful, here, or you may end up with an extension on files that you didn't intend.

PROTECT FILE

When you place a write-protect tab on a disk, nothing can be written to ANY file on that disk. While that is often an important step to take in order to protect commercial applications software or game disks, it can be inconvenient on disks that you work with regularly. Write-protect tabs, the hardware approach, is an all or nothing proposition. With the PROTECT FILE option, you may lock one file on a disk, or several files, or, even, all files.

> While PROTECT FILE provides a substantial amount of protection for your valuable files, IT IS NOT FOOLPROOF! Should you accidentally FORMAT a disk containing files--protected or not--those files will be WIPED OUT IRRETRIEVABLY by the FORMATting process.

With the dire warnings out of the way, the PROTECT FILE function is simple to use. To protect a single file, type "P", followed by the full filespec of that file. As an example:

VALUABLE.TXT <RETURN>

When you check the directory of the disk, there will be an asterisk to the left of the filespec on files that are protected.

To protect all the files on a disk, you may invoke wildcards. Type "P", and:

2:= <RETURN>

Here, the files to be protected were on drive 2. The point is that if you are working on any drive other than drive 1, you must specify the drive number you wish to have the computer work on. In any case, notice how easy it was to protect all of those files. The wildcard equals sign referred to every single file on the disk.

Selective protection can also be employed. A group of files with the same file name, but different extenders could be protected by typing "P" and:

NAMES.* <RETURN>

On the other hand, a group of files with different file names, but common extensions, could be protected by typing "P" and:

*.LST <RETURN>

In this example, all files bearing the LST extension are protected.

Protected files cannot be modified, updated, or deleted. They can, however, be copied to other disks, or to the same disk under another name. Should you copy a protected file to another location, the protection will not copy with the file. The only instance in which such protection will transfer is with the WHOLE DISK COPY "W" function (see pp. 28-29).

UNPROTECT FILE

Just as you need to be able to protect certain files, there are times when you need to unprotect them, as well. The UNPROTECT FILE option permits you to do just that, so that files can be updated or modified.

All of the instructions in the section on PROTECT FILE are valid with this option. That is, you may remove the electronic protection from an individual file, from several files, or from all files on a disk. Simply refer to the protocols listed above, remembering only that to invoke the UNPROTECT FILE option requires you to type a "U".

DELETE FILE

With this option you may remove one, several, or all of the files on a disk. DELETE FILE is, as the name indicates, a very destructive option, and must be treated with GREAT CARE. You can easily wipe out the contents of an entire disk if you are not very, very careful.

The DELETE FILE function erases the directory entry of the files you no longer want or need on your disks. In addition, it modifies the VTOC (Volume Table of Contents) sector just preceding the directory, thus freeing the sectors occupied by that file for use by the disk. The DELETE FILE function does not remove the files themselves from the disk. Instead, those files will be overwritten any time a new file is SAVEd or LISTed to the disk and the drive decides to put the new file onto the sectors occupied by the old file.

This approach; deleting the directory, modifying the VTOC, and leaving the files sectors themselves intact, suggests that there may be a way to recover deleted files if you have deleted them in error. Indeed there is such a way--provided the file sectors have not already been partially or wholly overwritten.

Reconstructing directories and VTOCs, however, are tasks that require advanced programming techniques which are beyond the scope of this user's manual. Please refer to the Bibliography at the back of the manual for books that may be of help with this problem.

To delete a single file, all that is necessary is to type "D". Immediately, the screen will turn red, and the prompt will read:

DELETE:FILE NAME?

Respond with the name of the file you wish to have deleted:

KAPUT.FIL <RETURN>

The computer will then echo the filename to you and instruct you to press "X" to delete:

FILE NAME D:KAPUT.FIL
"X" TO DELETE

Pressing any other key but "X" will abort the operation and leave your file intact.

With DELETE FILE the "query" function is "on" by default. That is, every time you seek to delete a file or files, the computer will ask you to confirm your desire by pressing the "X" key. If you have a lot of files to delete, or are VERY SURE which files you wished deleted, you may turn the query function "off" by appending a "/N" to the filespec. Thus, typing "D", followed by:

KAPUT.FIL/N <RETURN>

will bypass the confirmation request and merrily delete your file with no further intervention on your part.

As with many of the menu options, DELETE FILE can be used with wildcards. Since the preceding sections of this chapter feature numerous examples of the use of wildcards to aid in dealing with multiple files, one example here should serve to illustrate their use with this option.

Occasionally you may find that you have a disk or disks which contain files that you no longer need. There are two ways by which you can rehabilitate those disks to current use. You can, of course, reFORMAT them. However, you can also delete all the files on the disk(s) and PRESERVE the current formatting.

To accomplish this, type "D", followed by:

2:=/N <RETURN>

In this example, the disk in question is in drive 2. The equals sign tells the computer that all files are to be considered, and the "/N" turns off the query function so that you do not need to type "X" in order to delete each file. When the operation is finished, check the directory of that disk. You will get a report that there are 707 FREE SECTORS.

The safest way to practice this procedure is to copy a disk with several files on it, then try the technique ON THE COPY after putting the original disk away.

WHOLE DISK COPY

This option permits you to make an exact copy of a disk. It uses a technique called "sector copying." The purpose of WHOLE DISK COPY is to permit you to make backups of your important data and program disks. It is NOT intended for use in pirating copyrighted software. Most commercial software employs copy-protection of one kind or another in order to discourage theft.

Since this function copies disks sector by sector, it may be used only when both drives are configured to the same density: Single to single--double to double. To duplicate a disk in one density to a disk in the other density necessitates the use of the COPY FILE function.

If you plan to use this function with more than one drive then please verify that the destination drive is set to the same density as the source drive. If it isn't then RECONFIGURE (see pp. 33-34) the errant drive. Then, type "W". The screen prompt will read:

COPY WHOLE DISK
source drive # ?

Answer the prompt with the number of the drive containing the disk you wish to copy, say, "1". Next, the screen will ask:

DESTINATION drive # ?

Respond with a "2". The screen will now read:

"E" TO EXIT;ANY OTHER KEY TO CONTINUE

Before pressing any key, check to be sure that your source disk is in drive 1 and your formatted destination disk is in drive 2. When ready, press any key but "E".

The screen will turn green, and you will be presented with a status report. Whenever the computer is writing to the destination disk, the screen will switch to red. Upon completion, the screen will report:

currently on SECTOR:720

SECTORS READ:720 SECTORS WRITTEN:713

ERROR SECTORS:000 EMPTY SECTORS:007

DONE !

NO ERRORS

PRESS CHOICE OR <RETURN> FOR MENU

The numbers in the example above are for illustration only. The report you get may provide you with different results.

In order to copy a whole disk to another disk using one drive, then, after typing "W", answer the source, destination prompts with the same number--1,1. From there the computer will instruct you as to when to swap your disks. Should you need to use this option, please remember the warning issued earlier! Place a write-protect tab on your source disk so that errors will not produce disaster.

An important time-saving feature of SMARTDOS is that the WHOLE DISK COPY option will skip over empty sectors and "bad" sectors (sectors with errors), thus copying the disk in the minimum time possible.

KOPY SECTORS

This option is intended for use by experienced programmers. If you are a casual user or a new disk drive user, then you can easily skip this section. It is called KOPY SECTORS to distinguish it on the menu from the COPY FILE option.

The principal function of this option is to permit SMARTDOS users to salvage damaged disks by copying correct sectors from them to new disks, whereupon files can be reconstructed. With this option you can copy individual sectors from one disk to another with the same ease as copying files.

Disks can be damaged (electronically) in an almost infinite variety of ways; Unexpected power failures can send voltage "spikes" through the read-write head(s) of your disk drives; you can put a disk in the drive before turning on the power, or, conversely, turn the power off before removing your disk(s); your disk drives, or your computer itself, can malfunction. In short, if you use disks, then eventually you are going to need a feature such as the KOPY SECTORS option of SMARTDOS. Then, too, disks themselves do wear out, and when they do, copying the good sectors to another disk will often save you countless hours of reconstruction time.

Should you need to use this function, then your first task is to determine the sector numbers of the sectors you need to copy. Once you have made that decision you are ready to proceed. This function works best with two drives, but if you like to swap disks it will work with one drive, too. To use it with one drive simply specify that drive number in response to both the source disk and destination disk prompts. From there the computer will tell you when to insert the source disk and when to insert the destination disk.

For an operation involving two drives, the procedure is to first type "K". The screen message will read:

COPY SECTORS

Enter first sector

Respond with the number of the first sector you wish to copy, say 360 <RETURN>. The message will now read:

Enter last sector

Respond with your last desired sector, perhaps 368 <RETURN>. With that the computer will ask you for the:

Source drive # ?

Answer with a "1". Now the computer asks for the:

DESTINATION drive # ?

Type "2". Finally, the computer will instruct you to type:

"E" TO EXIT;ANY OTHER KEY TO CONTINUE

If you press any other key than "E", the procedure will begin, the screen will turn green while the source drive is reading sectors, then red during writes. The counters will indicate what sectors have been read and written. When complete, the computer will report to you that it is:

DONE !

NO ERRORS

PRESS CHOICE OR <RETURN> FOR MENU

In the example given above, you copied the VTOC and the Disk Directory from the disk in drive 1 to the disk in drive 2. By itself, that is not a very useful thing to do, but it does serve to illustrate the selective copying ability of this copy sector function.

Referring to the explanation above, if, when you are asked for the number of the first sector, you respond by pressing <RETURN>, the DOS understands that to mean that you want all the sectors from 1-720 copied. Doing that, by the way, is exactly the same as using the WHOLE DISK COPY function.

If you are copying sectors from a damaged disk to another disk, then no doubt the sector status report will indicate some sectors on which the computer detected errors. When that happens, the computer will finish the copying process. At that point the screen will read (the numbers you get will probably be different from those in the example):

```
                Currently on sector:720

SECTORS READ:704        SECTORS WRITTEN:591

ERROR SECTORS:016       EMPTY SECTORS:113

  DONE !

  "E" TO EXIT;ANY OTHER KEY TO CONTINUE
```

To review the errors, press any key but "E". You will be presented with a sub-menu:

```
                Disk Drive Speed:____ RPM

                                        Drive #2

PRESS:
     C=check drive speed
     L=list error sectors
     P=print error sectors
     Z=zero error sectors
     E=exit to main menu
```

The "C" option permits you to check the speed of your drives. The speed of your disk drives can have an adverse affect on their ability to read and/or write data correctly. If either of your drives is running too fast or too slow, those variations can produce errors.

The KOPY SECTOR Submenu item "C" will automatically check the speed of the source drive (the one from which copies of sectors are being made), but you can check the speed of any other drive at this time by pressing the drive number key for that drive, then, when the computer has accepted that keystroke (this will take a second or two), press the "C" key again.

To stop the speed test, press any key. At this point you will be able to select another item in the Submenu.

"L" will cause the computer to list for you all of the sectors on which errors were detected, and will tell you--briefly--what the errors were, and the error numbers that correspond. For example:

```
Sector 19:DEVICE DONE-144
Sector 20:DEVICE DONE-144
Sector 21:DEVICE DONE-144
```

"P" sends a listing of the error sectors to your printer. If there are many error sectors on a disk this can be a particularly handy feature.

The "Z" item permits you to have the computer write a pattern of zeros to all of the bytes in each error sector. This re-establishes continuity on the disk, so that the good sectors on the disk can be dealt with.

Finally, the "E" item returns you to the main menu.

TEST SECTORS

Occasionally you may experience problems with your disks. Programs may not load, or if they do load may not run correctly. With this option you may check that disk directly to determine whether the problem might be due to bad sectors. TEST SECTORS is a diagnostic tool which will enable you to examine a disk sector by sector, testing the integrity of each. After it has been run, you will be presented with the same Submenu that was described above. All of the items are the same in this mode as they were in the KOPY SECTORS option.

To run TEST SECTORS, insert your suspect disk into one of your drives, press the "T" key, and respond to the prompts. If, when asked for the number of the first sector, you press <RETURN>, the DOS will prepare to examine all of the sectors on the disk (1-720). The Source drive is the drive in which you put the disk to be examined.

FORMAT DISK

This is a very basic and fundamentally important function. New disks cannot be used until and unless they have been formatted. Disks that have "crashed" (are unusable) for one reason or another likewise may need to be formatted before they can be used again for new files and programs.

Formatting is the process by which a disk is prepared electronically by the disk drive. The disk is divided into its forty tracks, each of which is then subdivided into eighteen sectors. During the formatting process, the directory sectors are established, as is the VTOC sector. Thus, although each disk contains 720 sectors, by the time the disk is formatted, only 707 sectors are available for your use in storing data. That is 707 sectors in either single or double density.

To format a disk, first boot your system with SMARTDOS. If BASIC is active and you get a READY prompt, then type "DOS" to get to the menu. From the menu, type "F". The screen will turn red as a signal to you that you are invoking a function that requires care. The message that appears will read:

FORMAT:DISK NUMBER?

Respond with the number of the drive containing the disk you wish to format, for example, "2". When you do, a new message will appear:

PUSH "X" TO FORMAT DISK 2

Pressing any other key than "X" will abort the operation. Pressing the "X" key will cause formatting to begin. Before you format a disk, check the density status report at the top of the menu to verify that the drive in which you are going to format the disk is configured for the density you desire. If it is not configured correctly, then use the RECONFIGURE option (see below) to put the drive into the proper density.

> CAUTION: Formatting a disk will destroy all existing data, files, and programs on that disk. Be very sure that you have inserted the correct disk before you press the "X" key.

RECONFIGURE/ON

This menu option permits you to reconfigure the density status of any or all of your drives at will. Since SMARTDOS is "smart," it will permit your disk drives to sense the density of any disk inserted into a drive, and reconfigure themselves to the appropriate density for that disk. Because of that powerful feature, you may only need to use the RECONFIGURE function when you are building new disks. For instance, if you wish to make a double density copy of, say, SMARTDOS, (and you have two drives) then you would need to RECONFIGURE drive 2 to double density first. To do that, type "R". You will get the message:

NEW DRIVE OR DENSITY:PRESS DRIVE #?

Press the "2" key to reconfigure drive 2 to double density.

Once you have done that and have formatted a disk in double density, then you can MAKE SYS FILES ("M"). Following that, you may proceed to COPY the other files onto that disk. The one file on the distribution disk that won't be copied by this procedure, assuming that you have used wildcards to COPY the other files, is AUTORUN.SYS. To copy that file to your new double density version of SMARTDOS requires you to COPY it by using its name. To run through that procedure one more time, type "C" (for COPY FILES), and:

AUTORUN.SYS,2:AUTORUN.SYS

With this one file, you must type out its name both in the Source designation AND in the destination. Wildcards will not work.

If you have one disk drive, the business of copy a disk in one density to a disk in another density is very simple, if rather tedious. To use the same example as above (making a double density copy of SMARTDOS), first FORMAT a disk in double density. Next, place your single density SMARTDOS disk back in your drive and type "M". When you are asked for a disk number, answer by typing "1". Before pressing the "Y" key, be sure you have removed your source disk from the drive and have replaced it with the formatted double density disk.

Once again, insert your source disk into the drive and type "C". When you get the prompt: "COPY:SOURCE,DEST", type "=" <RETURN>. That is the equals sign followed by the <RETURN> key. From there, the computer will instruct you as to when to insert the source disk and when to insert the destination disk. When the source disk should be in the drive the screen will be green, and when the destination disk should be in the drive, the screen will turn red. On the SMARTDOS disk, all of this will entail disk swaps for every file to be copied (3 files with the wildcard). Finally, press "C" again and answer the prompt with "AUTORUN.SYS"<RETURN>. To remind you, the COPY FILE function will not copy any SYS files.

When you are finished, check the directory of that disk. You will find that you have a SMARTDOS disk with 624 free sectors on it! That contrasts with the paltry 544 sectors available on a single density version of the SMARTDOS disk.

If you haven't worked with double density disks before, you will be positively amazed to learn how much data you can store on one.

Perhaps you noticed something odd about the prompt message you got when you pressed "R" to invoke the RECONFIGURE/ON function (maybe you also wondered about the /ON at the end of RECONFIGURE). The message you got, if you recall, was:

NEW DRIVE OR DENSITY:PRESS DRIVE #?

The purpose of the "/ON" and the NEW DRIVE messages is to permit you to bring additional drives "on line" even if they weren't on when you booted your system! Suppose you have additional drives connected to your system, but had refrained from turning them on. During the course of your computing session you suddenly realize that you need another drive. With other DOSs, you would be out of luck. With SMARTDOS, though, all you have to do is turn your other drive(s) on and, assuming that they are configured to be drive numbers other than one, invoke the RECONFIGURE/ON function. When you get the prompt message, respond with an appropriate number and SMARTDOS will place those drives on line for you without the necessity to reboot the whole system.

OBVERT RESIDUP

This is a feature of SMARTDOS that places it head and shoulders above any other DOS for ATARIs. This option allows you to decide whether you want DUP.SYS (the Disk Utilities Programs) to remain resident in RAM or not. The "normal" status of DUP.SYS is that it is "non-resident." In order to give you the largest possible amount of RAM for your programs, DUP.SYS is brought into memory only when you need it by typing "DOS."

If you are involved in a heavy programming session, though, you may find it extremely useful to have DUP.SYS in memory all the time. When it is "resident," and you type "DOS" to return to the menu, the program you are working on remains completely undisturbed. When you have done what you need to do from DOS, you simply type "E" to return to BASIC and resume your programming without pause.

There is, of course, a trade-off here. You can have either maximum memory available to you, or you can have maximum convenience. To give you an idea of the costs involved in these choices, some statistics are in order. In a 48K ATARI 800, the amount of free RAM available to you when RESIDUP is "off" is 32,246 Bytes.

With RESIDUP "on," that drops to 22,639 Bytes. These figures, by the way, are exactly what you will get if you are running SMARTDOS on a "64K" 800XL!!

When you type "O" to invoke OBVERT RESIDUP, notice what happens to the RESIDUP status report near the top of the menu. It will tell you at a glance whether RESIDUP is on or off. This function is a "toggle." That is, it switches back and forth between two states, exactly like a toggle switch. As distributed, the default status of RESIDUP is off. During a programming session, if you turn it on and leave it on, then remove your DOS disk, you will find that the next time you boot with that disk, RESIDUP is on. That is nothing to get alarmed about, for you can change its status any time you wish.

VERIFY/RETRY

When you wish to SAVE or LIST files to disk, there are essentially two ways to accomplish that important task. One way is to have the computer and the disk drive--both of which are "intelligent"--verify the accuracy of the process at the time the data are transferred to the disk. While this approach, which is standard in ATARI DOS, assures virtually complete accuracy in data transfer, it is relatively slow. Actually the computer is performing two tasks; writing and then verifying the accuracy of its writes. Should it detect an error in the checksums it uses, it will report the error to you. The second approach, which is the default in SMARTDOS, is to have the computer write files to disk without verifying the accuracy of the write. This is considerably faster than the former approach. But, you may ask, isn't this risky? Not really.

Data transfer between ATARI computers and their peripherals is inherently accurate. The number of instances of errors creeping into the file transfer process will be miniscule, indeed, unless there is something seriously wrong with either your computer or your disk drive(s). SMARTDOS chose to take advantage of the very low error rate by leaving VERIFY "OFF." However, in keeping with the philosophy of providing the user with as wide a range of choices as possible, SMARTDOS allows you to turn VERIFY "ON" any time you want it on. For particularly critical files, this is recommended.

This function, like RESIDUP/ON, is a toggle. Should you wish to turn VERIFY/RETRY on, press the "V" key. You will read the message:

Select # error retries; verify (Y/N)
RETRY 0 TIME(S); WRITE VERIFY:OFF

Press the "2" key, for example, to instruct the computer to retry twice before reporting an error. Then press the "Y" key to turn VERIFY on. Finally, press the <RETURN> key. When you do that, the menu will refresh itself. The status box toward the top of the menu will report that:

VERIFY:ON

Whenever you wish to speed up the writing process, you may turn VERIFY/RETRY off by again pressing "V". Now the message will read:

Select # error retries; verify (Y/N)
RETRY 2 TIME(S); WRITE VERIFY:ON

This time, press "Y" (with that "N" followed by SETTING). After menu refresh, the status box will reflect that TERSE is OFF.

LOAD FILE

This function is designed for use by experienced programmers. It permits the user to load "binary" files from DOS. A binary file can be data, but it is more often a machine language program of one kind or another. In any event, a binary file cannot be loaded into memory from BASIC.

There are several binary files included on your SMARTDOS disk. They include DOS.SYS, DUP.SYS, RESIDENT, DEFAULT, and ATTORNEY.SYS. Of those, the only one that can be called from BASIC is, of course, DUP.SYS. When you type "DOS" from BASIC, the command is vectored through DOS.SYS (which is always in memory) to a machine language subroutine that loads DUP.SYS.

To load any other binary file you must be at DOS (the menu), and you must use the LOAD FILE option. To illustrate this function, let's use the binary file called DEFAULT. You need to run this program anyway in order to "customize" your copy of SMARTDOS. First, type "L". The prompt will read:

LOAD FILE NAME?

respond with the name of the file. Type "DEFAULT." When the file has loaded, the menu will scroll up if the screen is replaced with the first page of the DEFAULT program. First, the program will report to you on all of the current values. Drives 1 through 4 will have buffers; there will be six LOG byte buffers, file I/O timeout/retry will be default, and the autorun search range will be set from $A2E0 through $A2FF. At the bottom of the screen the message will read:

Do you wish to change these ? (Y/N)

these defaults are satisfactory to you, then press "N", and you will be returned immediately to the menu. If you wish to change any of the defaults, then press "Y" and follow the prompts.

The reason to consider changing defaults is to save memory. If you have, for example, two disk drives then why provide buffers for drives 3 and 4? The default of six I/O buffers is the minimum number necessary to support disk drives, and will probably be quite satisfactory for you unless you do an unusual amount of file handling. The $A2E0 range can certainly be changed. If you wish to have DOS automatically load and run a series of programs for you before turning control of the computer over to you, then increase the number appropriately from default of 2.

If you attach a ".M" to the end of the filespec you specified in the LOAD option, the computer will load the file, but it will not be run. On the other hand, ".P" to the filespec will allow the file loaded to run, but only if the file contains JMP or RTS addresses (see SELECT I/O, below).

BINARY SAVE

BINARY SAVE is another advanced function. Before using it you should consider familiarizing yourself with assembly language and hexadecimal numbers.

With this option you may save blocks of memory (RAM only) to disk. When you use the ATARI EDITOR/ASSEMBLER cartridge (or some other assembly language editor), you will need to save the results of your programming efforts to disk. This function enables you to do just that.

When you invoke this option by pressing "B", you will be prompted to:

SAVE:FILE NAME,START,END(,INIT,RUN)

Here, you are expected to assign a name (filespec) to your file, and to indicate to the computer where it can find the STARTing point in memory for that file, its ENDing point, and, optionally, an INITialization point and/or a RUN address--all in hexadecimal notation.

Additionally, you may append memory blocks to existing binary files by using the "/A" option after the filespec. Be advised again that creating these "compound" files requires an intimate familiarity with assembly language, hexadecimal (base 16) numbers, and the potential problems involved in chaining binary files together. For further information on these topics, please consult the Bibliography in the back of this manual.

GO TO ADDRESS

This option permits you to specify an address--in hexadecimal--from which you wish to have a machine language program (previously LOADed into memory) begin to run. It also permits you to send the computer to a specified address in ROM that may contain a vector to a machine language subroutine and to run that subroutine. While this is a very powerful feature; it, too, requires care in its use, lest you unwittingly send the computer off into never-never land; requiring you to turn the system off, then on again to recover. If that happens, you will lose whatever program you may have had in memory. You won't, however, do any damage to the computer.

SPEED CHECK

Periodically, it is wise to check on the speed of your drives. As was mentioned above, excessive variations from the nominal speed may cause problems both in reading and writing data. If any of your drives are running too slow or too fast, or are fluctuating in speed, then take your unit in for servicing.

To use this function, type "S" and follow the prompts.

EXIT SMARTDOS

The principal function of this option is to permit you to leave the DOS menu and return to BASIC, or whatever cartridge you may be using (such as the ATARI EDITOR/ASSEMBLER cartridge).

If RESIDUP is on when you execute the EXIT SMARTDOS option, then you can go back and forth freely between BASIC and DOS with no time (or memory) lost. If RESIDUP is off when you EXIT SMARTDOS, then typing "DOS" will reload DUP.SYS into memory from disk. Thus, any program you may have had in RAM will be lost unless you had first SAVEd it to disk before typing "DOS."

BIBLIOGRAPHY

This annotated Bibliography is included for those of you who wish to delve more deeply into the fascinating innards of your ATARI computer. The books listed below by no means constitute an exhaustive list of the publications available. Instead, they are a fairly representative sample which will provide a very sound "jumping off place" for inquiry.

The works listed below are all books. But many of the most insightful explorations into ATARI computers have been published as articles or columns in various computer magazines. Over the years, for example, the column appearing in Creative Computing, called "Insight, ATARI;" especially the columns written by David and Sandy Small, as well as those penned by John Anderson, are well worth looking up. ANTIC: the ATARI Resource, San Francisco, CA, and ANALOG Computing: The Magazine for ATARI Computer Owners, Cherry Valley, MA, are veritable gold mines of information for ATARIans. Both of these fine magazines are published monthly.

So, what are you waiting for? Get on with it! Dig in and find out what makes these terrific ATARIs tick!

General Interest

Crawford, Chris, and others, De Re ATARI. Sunnyvale, CA: ATARI, Inc., 1981.

Chris and his brilliant crew have created a veritable Tour de Force with this book. No serious Atarian can afford to be without it.

Disk Operating System II Reference Manual. Sunnyvale, CA: ATARI, Inc., 1981.

The DOS 2 manual describes, in what is occasionally a clear fashion, the DOS that is the basis for all Disk Operating Systems for ATARI computers.

ATARI BASIC Reference Manual. Sunnyvale, CA: ATARI, Inc., 1980.

This is the manual that used to come with ATARI computers. And, although it is frequently even more abstruse than the DOS 2 manual, it is still an essential source of information about the computers as well as about ATARI BASIC. Do watch out for errors, though: this manual is littered with them.

ATARI Home Computer System Technical Reference Notes. Sunnyvale, CA: ATARI, Inc., 1982.

This is really three volumes of information in one package. Included are the Operating System User's Manual, the Operating System Source Listing, and the Hardware Manual. There is no other source of detailed information about every aspect of ATARI computers more complete and authoritative than this publication. ATARI is to be commended for making this available to those who use their computers. While the version I have was written before the XL computers were introduced, there may be a revised edition available. In any [illegible] for your library.

Carris, William, Inside ATARI BASIC: A Fast, Fun, and Friendly Approach. Reston, VA: Reston Publishing Company, Inc., 1983.

For those of you who are new to this strange computer language known as BASIC, this is a very lighthearted introduction to it. Filled with examples, it will ease the task of learning a bit about the intricacies of programming a computer.

Chadwick, Ian, Mapping the ATARI. Greensboro, NC: COMPUTE! Books, 1983.

A very thorough examination of the functions of the many memory locations within your computer. There are several "memory maps" available for ATARI computers, and this is probably the most comprehensive and complete of all of them.

Poole, Lon, and others, Your ATARI Computer: a Guide to ATARI 400/800 Computers. Berkeley, CA: OSBORNE/McGraw-Hill, 1982.

This is the book that should have been shipped with every ATARI computer. We can hope that the authors will soon produce a second edition covering the new XLs, as well. This is a well-written examination containing a wealth of valuable hints, tricks, and techniques for squeezing the most from your computer.

Wilkinson, Bill, Inside ATARI DOS. Greensboro, NC: COMPUTE! Books, 1982.

Bill Wilkinson's modesty necessitated that he be called the "compiler" of this book. It is the definitive resource for ATARI-like DOSs, written by the people who wrote the original! The real jewel of this slim volume--not to detract from the lucidity of the text--is the complete Source Code listing of ATARI DOS 2. Don't pass this one by.

Assembly Language

Mansfield, Richard, Machine Language for Beginners. Greensboro, NC: COMPUTE! Publications, Inc., 1983.

Written for people who have a passing knowledge of BASIC, and are curious about "machine" language. Actually, the book is all about Assembly language, but who's counting? Author Mansfield apparently is a born teacher, for he takes the reader easily through a very technical topic with humor and grace. And, though the book is designed for anybody who uses a computer employing the 6502 microprocessor chip (the heart of the ATARI), it contains specific examples and references to ATARIs, as well as to the other (lesser) computers it covers. If you have been curious about hexadecimal numbers (and who hasn't been?); if you have wanted to overcome the speed limitations of BASIC, then this terrific book is just for you.

Inman, Don, and Inman, Kurt, The ATARI Assembler. Reston, VA: The Reston Publishing Company, Inc., 1981.

While this little book was designed principally for those who use the ATARI

Assembler Cartridge, it, too, is a good introduction to Assembly Language programming.

Leventhal, Lance A., 6502 Assembly Language Programming. Berkeley, CA: OSBORNE/McGraw-Hill, 1979.

This one is for those of you who are REALLY SERIOUS about learning Assembly Language. Frequently thought of as the definitive work on the topic, Leventhal's book is well worth the effort required to understand it. There is very little about the capabilities of the 6502 microprocessor that isn't explored in great detail in this masterwork.

LIMITED WARRANTY

This software product and the attached instructional materials are sold "AS IS", without warranty as to their performance. The entire risk as to the quality and performance of the computer software program is assumed by the user. The user, and not the manufacturer, distributor or retailer assumes the entire cost of all necessary service or repair to the computer software program.

However, to the original purchaser only, THE PROGRAMMERS WORKSHOP warrants that the medium on which the program is recorded will be free from defects in materials and faulty workmanship under normal use and service for a period of ninety (90) days from the date of purchase. If during this period a defect in the medium should occur, the medium may be returned to THE PROGRAMMERS WORKSHOP or to an authorized THE PROGRAMMERS WORKSHOP dealer, and THE PROGRAMMERS WORKSHOP will replace or repair the medium at THE PROGRAMMERS WORKSHOP's option without charge to you. Your sole and exclusive remedy in the event of a defect is expressly limited to replacement or repair of the medium as provided above. To provide proof that you are the original purchaser, please complete and mail the enclosed Owner Warranty Card to THE PROGRAMMERS WORKSHOP.

If failure of the medium, in the judgment of THE PROGRAMMERS WORKSHOP, resulted from accident, abuse or misapplication of the medium, then THE PROGRAMMERS WORKSHOP shall have no responsibility to replace or repair the medium under the terms of this warranty.

The above warranties for goods are in lieu of all other express warranties and no implied warranties or merchantability and fitness for a particular purpose or any other warranty obligation of the part of THE PROGRAMMERS WORKSHOP shall last longer than ninety (90) days. Some states do not allow limitations on how long an implied warranty lasts, so the above limitation may not apply to you. In no event shall THE PROGRAMMERS WORKSHOP or anyone else who has been involved in the creation and production of this computer software program be liable for indirect, special, or consequential damages, such as, but not limited to, loss of anticipated profits or benefits resulting from the use of this program, or arising out of any breach of this warranty. Some states do not allow the exclusion or limitation of incidental or consequential damages so the above limitation may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

The user of this product shall be entitled to use the product for his/her own use, but shall not be entitled to sell or transfer reproductions of the product or instructional materials to other parties in any way.

NOTICE:
THE PROGRAMMERS WORKSHOP reserves the right to make improvements in the product described in this manual at any time and without notice.